Plan-graph Based Heuristics for Conformant Probabilistic Planning

Sailesh Ramakrishnan
University of Michigan
and NASA Ames Research Ctr.
Moffett Field, CA 94035
sailesh@email.arc.nasa.gov

Martha E. Pollack
Dept. of Computer Science
University of Michigan
Ann Arbor, MI 48109
pollackm@eecs.umich.edu

David E. Smith
Computational Sciences Division
NASA Ames Research Ctr
Moffett Field, CA 94035
de2smith@email.arc.nasa.gov

Abstract

In this paper, we introduce plan-graph based heuristics to solve a variation of the conformant probabilistic planning (CPP) problem. In many real-world problems, it is the case that the sensors are unreliable or take too many resources to provide knowledge about the environment. These domains are better modeled as conformant planning problems. POMDP based techniques are currently the most successful approach for solving CPP but have the limitation of state-space explosion. Recent advances in deterministic and conformant planning have shown that plan-graphs can be used to enhance the performance significantly. We show that this enhancement can also be translated to CPP. We describe our process for developing the plan-graph heuristics and estimating the probability of a partial plan. We compare the performance of our planner PVHPOP when used with diffrent heuristics. We also perform a comparison with a POMDP solver to show over a order of magnitude improvement in performance.

Introduction

In this paper, we develop plan-graph based heuristics to address a variation of the conformant probabilistic planning (CPP) problem. Informally, the conformant planning problem is the problem of finding an unconditional sequence of actions that guarantees the successful achievement of the goal after executing that sequence in an environment that is non-observable. The probabilistic version (CPP) maximizes the probability of the goal being true. Actions have preconditions and effects which can be conditional. CPP is the conformant planning problem with numeric probabilities for the effects of actions. In this paper we focus on a variation of CPP where the objective is to find a plan that has a probability of achieving the goal that is above a particular threshold, similar to Buridan (Kushmerick, Hanks, & Weld 1995) and Tgraphplan (Blum & Langford 1999).

In many real-world problems, it is the case that the sensors are unreliable or take too many resources to provide knowledge about the environment. These domains are better modeled as conformant planning problems rather than having to model the complicated sensors. It may be useful however to find a plan that has a reasonably good chance of achieving the goal rather than spending a lot of computational effort to find the plan that is most likely to succeed.

Previous approaches to CPP can be broadly classified into MDP-based and other (non-MDP) approaches. Non observable MDP's (NOMDP's) can represent CPP and currently are the most successful approach. A popular POMDP solver pomdp-solve(Cassandra) which can also solve NOMDPs (since they are a degenerate case of POMDPs) has outperformed other approaches without sacrificing the optimality of the solution found. Other approaches include the probabilistic versions of Partial Order Causal Link (POCL) planning (CBuridan(Draper, Hanks, & Weld 1994), Mahinur (Onder & Pollack 1999), Weaver (Blythe 1998)), SAT Planning (ZANDER(Majercik & Littman 1998)), GraphPlan (Pgraphplan and Tgraphplan(Blum & Langford 1999)) and more recently Constraint Satisfaction (CSP) techniques (CP-Plan(Hyfil & Bacchus 2003)). These planners can also be categorized based on whether they use a state-space representation (pomdp-solve, ZANDER, CPPlan), a plan-space representation (Buridan, Mahinur, Weaver) or an abstracted state-space representation (Pgraphplan and Tgraphplan).

There has also been a lot of research in conformant planning (without quantitative probabilities). Notably, Bryce and Kambhampati(Bryce & Kambhampati 2003) have shown that plan-graph based heuristics can make a tremendous impact in the scalability of such planners. Their results show atleast one order of magnitude improvement in the time taken to solve the conformant planning problem compared to CGP(Smith & Weld 1998) and MBP(Bertoli *et al.* 2001).

In this paper, we present a new probabilistic POCL planner called PVHPOP which perfoms significantly better than other CPP planners. The key enhancement to previous POCL approaches is the use of plan-graph based heurisitics. PVHPOP is based on VHPOP(Younes & Simmons 2002), a POCL planner that performed favorably in the 2002 IPC.

PVHPOP searches the plan-space guided by a plan ranking heuristic based on a relaxed plan-graph. The relaxed plan-graph is used to provide an estimate of the probability of achieving an individual literal; these estimates are then combined to evaluate the probability of a partial plan. This probability estimate is then used in a ranking function to rank various partial plans which guides the search over plans.

This paper is organized similar to (Hyfil & Bacchus 2003), which compared CPPlan to state-of-the-art CPP planners. In the next section, we begin by motivating the use of plan-graphs for probabilistic planning, then introduce the PVHPOP algorithm and finally perform a comparison to the POMDP solver as well as CPPlan.

Motivation

Our motivation for using a plan-graph based heuristic is derived from the following observations. POCL based probabilistic planners have performed poorly in general due to non-informative heuristics that have not guided the planners to good plans. Further research in probabilistic POCL planning has enhanced the techniques for assessing and improving the probability of plans, but there has not been significant improvement in heuristics.

Planners based on MDP's use a state space based representation to identify a policy from states to actions that maximizes the expected utility. One main drawback of this approach is that the state space tends to become extremely large even for moderately sized problems. However algorithms such as value and policy iteration produce optimal policies, when sufficient time and space is available.

More recently, deterministic POCL planning has been re-emerging as a viable technique due to the use of better heuristics. Plan-graphs and plan-graph based heuristics have shown significant promise. In the AIPS 2002 planning competition, almost all the competitive planning systems used a plan-graph as a basis for deriving heuristics to guide their search. Even though all the domains in the planning competition were deterministic, this encourages further study of the use of plan-graphs to guide search in plan space.

In deterministic conformant planning, Bryce and Kambhampati showed that using plan-graph based heurisitics, the performance of CAltCAlt, a POCL planner can be improved by at least an order of magnitude. This result significantly encourages the use of plan-graph heuristics in probabilistic conformant planning.

Conformant Probabilistic Planning Problem

A probabilistic planning domain is a pair $\langle P, A \rangle$, where P is a set of propositions and A is a set of actions that operate on these propositions. These actions have uncertain effects each occurring with a particular probability. These effects can be conditional on the truth of other propositions.

The input to CPP is a 3-tuple $\langle S_i, G, t \rangle$, where S_i is a set of propositions that define the initial state, G is a set of propositions that define the goal state and t is a probability threshold ($0 \leq t \leq 1$). Solving a CPP problem involves finding a plan (an unconditional sequence of actions) that has a probability greater than or equal to t of reaching a goal state from the initial state. What makes this planning problem conformant is that the environment is completely unobservable. Hence exogenous events as well as the effects of actions are not observable. Traditionally CPP has been defined as a maximization problem whereas in our variation, we define it is a threshold satisfaction problem.

Here is an example of an action description with the associated probability information:

```
(:action fast-welding
  :parameters (?p1 ?p2 - part)
  :cost 100
  :preconditions
    (or (and
          (not (painted ?p1))
          (not (painted ?p1)))
        (and
          (smooth ?p1)
          (smooth ?p2)))

  :effect
    (and
      (when (and (clean ?p1)
                 (clean ?p2))
            (prob 0.9
                (welded ?p1 ?p2)))
      (when (not (clean ?p1))
            (prob 0.5
                (welded ?p1 ?p2)))
      (when (not (clean ?p2))
            (prob 0.5
                (welded ?p1 ?p2)))
      (when (and
              (not (clean ?p1))
              (not (clean ?p2)))
            (prob 0.3
                (welded ?p1 ?p2)))))
```

The action of *fast-welding* has two parameters ?p1 and ?p2, namely the two parts to be welded. These parameters can be typed as shown in the example or untyped. The set of preconditions describe when this action can be performed which in this case is either when both the parts are unpainted or both parts are smooth. The effects which are conditional express the various conditions in which the two parts are welded with different probabilities. The various conditional sections have to be exhaustive to cover every situation that impacts the probability of the two parts being welded. Finally, the last 5 lines of the example describe the probability of asserting each of the 4 effects and the cost of this action based on the values of the parameters.

Using a Plan-Graph for Heuristics

A Plan-graph is typically a bi-partite graph with alternating proposition and actions levels. The graph starts with all the propositions that are true in the initial state. The next level, which is an action level, contains all the actions that can be performed given the propositions in the previous level. This is followed by another proposition level consisting of all the propositions made true at this level including any new propositions made true by the actions in the previous level. There are arcs between propositions and actions that need those propositions as preconditions. Similarly, there are arcs between actions and the propositions that are produced as effects of those actions. An example plan-graph is shown in figure 1. The ellipses represent propositions and the rectangles represent actions.

A mutual exclusion constraint or mutex is used to mark

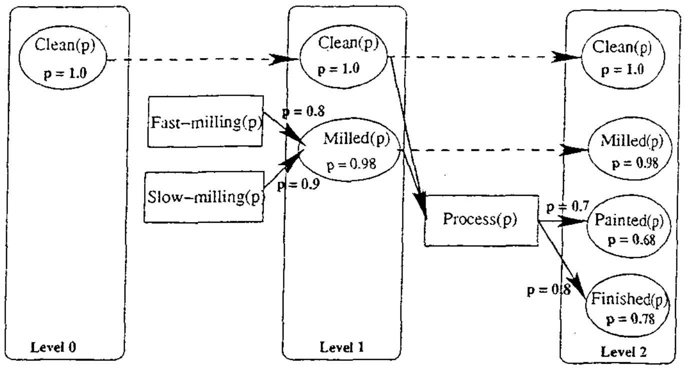

Figure 1: Estimating probabilities using a plan-graph

pairs of propositions that cannot be both true and pairs of actions that cannot be performed at the same time. In a relaxed plan-graph the propositions negated by actions are not added. Hence negative effects are not represented in this graph. The graph is usually developed until quiescence, i.e. until no new actions are applicable and no new propositions are generated.

Plan-graphs have been used to provide distance metrics. The level at which a proposition first appears is the minimum number of actions needed to assert that proposition. Similarly, an estimate of the effort needed to achieve a set of propositions can be made by summing the effort for each individual proposition. For example, in the plan-graph show in figure 1, the distance estimate for the propositions *clean(p)* and *painted(p)* is 2 (0+2).

Using the sum may be an overestimate since actions can produce more than one proposition as in the case of the *process(p)* action. Nonetheless, these estimates can be used to evaluate a partial plan and guide a planner in selecting and refining plans that minimize these distance estimates and hence more likely to lead to the goal.

Similar to assessing the amount of effort to assert a proposition, the probability of a proposition being true can also be propagation along a plan-graph by multiplying the probabilities of various effects needed to produce that proposition. Assuming independence, if there are two ways of getting proposition A each with probability p_1 and p_2 respectively, then the aggregate probability can be calculated as:

$$agg - prob = p_1 + p_2 - (p_1 * p_2) \qquad (1)$$

Again, since propositions can share common actions, an admissible estimate of the probability is to take the maximum, i.e. $max(p_1, p_2)$. Consider the example show in figure 1. *clean(p)* is present in the initial state and hence has a probability of 1.0. There are two independent actions (fast and slow milling) that can be used to assert *milled(p)*. Hence the aggregate probability of *milled(p)*, calculated using the formula given in equation 1 is 0.98. This probability can then be further propagated to derive 0.68 and 0.78 as the estimates for *painted(p)* and *finished(p)* respectively.

In a relaxed plan-graph, since there are no delete effects represented, each proposition, once it appears at a particular level, is repeated in every succeeding level. Similarly, actions can be repeated at each level as well. Therefore the probability of a proposition increases from the first level that it appears and asymptotically approaches 1.0. The probability estimates can use either the probability of a proposition at the first level or any level after. However, using a probability from a level higher than the first level could be an estimate based on actions needing to be repeated which, given that there are no delete effects represented, may not be feasible.

In this present work, the probability estimates for propositions are calculated without assuming that actions can be repeated. Additionally, multiple mechanisms of producing a proposition are assumed independent. Furthermore, mutexes are not considered. Because of these assumptions, the probability estimates do not definitely under or overestimate and hence are not an admissible heuristic.

PVHPOP

The algorithm underlying PVHPOP is similar to the basic POCL planning algorithm except for an additional step of creating a relaxed plan-graph in which probability estimates are calculated for all the propositions. The planning process starts with a partial plan consisting of a dummy INITIAL step whose effects are all the initial conditions and a dummy GOAL step whose preconditions are the goal propositions, as well as the initial unsatisfied open conditions. A particular open condition is selected and partial plans are produced which each incorporate a different way of supporting the selected open condition. Then, a particular child is selected as the next current plan and the process continues. Actions that modify propositions needed by other actions are considered threats(flaws) and have to be resolved. In the probabilistic version, a plan is deemed complete, if all the threats are resolved and all open conditions are at least supported once. When the plan is complete, a check is made to ensure that the plan has a probability greater than the threshold. If not, the search continues. This algorithm is described below:

```
PVHPOP($S_i$, $G$, $t$)
1. Create relaxed plan-graph and estimate probability of
   propositions.
2. Initialize: $curr - plan = S_i \cup G$.
3. While (Probability ($curr - plan \leq t$) do
4.     SELECT a flaw in $curr - plan$ to resolve.
5.     Generate children of $curr - plan$ by
       different flaw resolutions.
6.     SELECT a child to be $curr - plan$
7. endWhile
8. return $curr - plan$
```

The PVHPOP algorithm is different from the standard POCL algorithm in steps 4,5 and 6. In step 5, the children of the current plan are generated. There are two alternate ways of doing this. Some previous approaches (Mahinur for example) initially develop a plan to completion such that there are no threats and every open condition is supported exactly by one causal link. They then try to improve the quality of

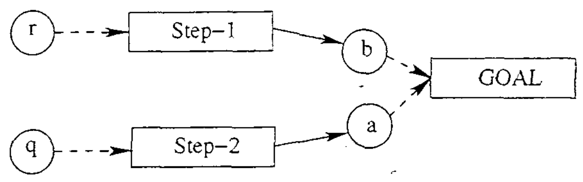

Figure 2: A partial plan

this plan by selectively reopening supported open conditions for additional support.

An alternative method is to create another child plan where the condition is left open in addition to being supported. For example, let proposition l be an open condition in plan $p1$ that has been selected for resolution. Then child plans $p2$ and $p3$ are created where l is resolved by adding a step and adding a link respectively. Additionally, child plans $p4$ and $p5$ are created. $p4$ is the same plan as $p2$ but additionally with a copy of l still left as an open condition. This provides the planner the ability during search to multiply support a proposition.

The SELECT in steps 4 and 6 represent non-deterministic choices. In this work, we focus only on the choice of the child plan (in line 6) using a combination of least refinements and most work involved as the heuristic for flaw selection. This plan selection is made by selecting the highest ranking child plan, where the ranking is a function of the probability estimate of the plan.

The probability estimate of the plan is calculated in the following manner. The propositions that are open conditions in the plan are assigned the probability as estimated from the plan-graph. These probabilities are then propagated forward towards the goal along the causal links in the plan. Consider the partial plan in figure 2. The circles represent propositions and the rectangles represent actions. The dashed lines represent open conditions and the solid lines represent causal effects. The goal propositions of a and b are represented as open conditions of a special GOAL action. From the plan-graph, p_r and p_q are estimated. The probability of b is then calculated as

$$Prob(b) = p_r * p_e^1 \qquad (2)$$

where p_e^1 is the probability of the effect of Step-1. Similarly, the probability of a is calculated as $p_q * p_e^2$. The final probability of the goal is then calculated as the product of the probability of a and b:

$$Prob(GOAL) = Prob(a \cup b) = p_r * p_e^1 * p_q * p_e^2 \qquad (3)$$

Since the actions are assumed to be independent, the different orders of execution of actions does not impact the probability estimate.

Plan Rank Heuristics

The estimate of the probability of a plan is used in a plan ranking function to guide the planner. We implemented two heuristics, the additive and the ratio heuristic. Both these heuristics balance the need to find plans with higher probability of success with the desire to find the plans that are efficient to execute.

Additive Heuristic

The additive heuristic for plan rank is given by the following formula.

$$PlanRank(p) = k * Prob(p) - Cost(p) \qquad (4)$$

Prob(p) is the probability estimate of the partial plan and Cost(p) is the sum of costs of all the actions in the plan. k is a weighting factor to balance the influence of the cost over the probability. While the value calculated for the plan rank using this heuristic does not have a clear semantic, it seems to perform well in practice.

Ratio Heuristic

The ratio heuristic was inspired by a paper by Simon and Kadane(Simon & Kadane 1975). They used the ratio of the value of goals over the sum of their costs to decide which set of goals to pursue. The ratio provides a feel for the increase in cost needs to increase probability. This heuristic is very sensitive to the cost/probability trade-off. However, it does not seem to perform very well when the focus is skewed to either only probability or only cost. This issue is further discussed in the experiments section. The ratio heuristic for plan rank is given by the following formula:

$$PlanRank(p) = Prob(p)/Cost(p) \qquad (5)$$

Experiments

PVHPOP was implemented by modifying the deterministic POCL planner called VHPOP (Younes & Simmons 2002). VHPOP is versatile planner that has support for many different kinds of heuristics including a relaxed plan-graph. The main planning algorithm was enhanced by the forward probability assessment algorithm while the plan-graph was modified to support the propagation of probabilities for each proposition. The planner was also configured to ground all actions.

Both the Additive and Ratio heuristics were implemented for use as a plan-ranking functions. The set of heuristics provided by VHPOP for flaw selection were left unchanged. These heuristics were set at a fixed setting for all experiments.

In order to test the performance of PVHPOP, a simple factory domain was created. It consists of parts that can be milled and ground. The milling and grinding actions come in two speeds: fast and slow. Hence there are 4 distinct action types. Problems of increasing size can be easily constructed by increasing the number of parts in the domain. The costs of the actions ranged between 100 to 200. The weighting constant for the additive heuristic was set at 10000. An important feature of this domain is the all actions are repeatable. The domain and the problems were represented in PDDL. The probability and cost information were stored in associated files and provided to the planner.

The first set of experiments were to compare the efficacy of the two different heuristics. The second set focused on a comparison between Cassandra's POMDP solver and PVHPOP. The POMDP solver is currently the best available for

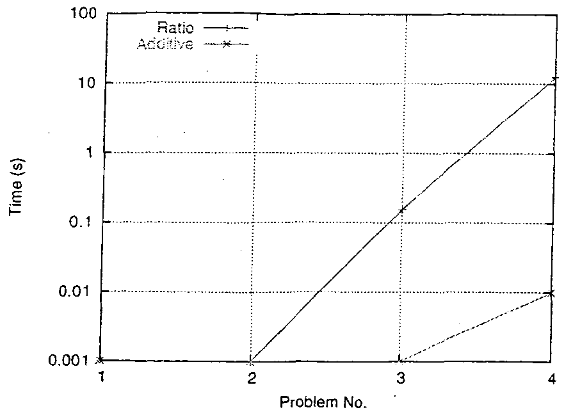

Figure 3: Additive versus ratio heuristic

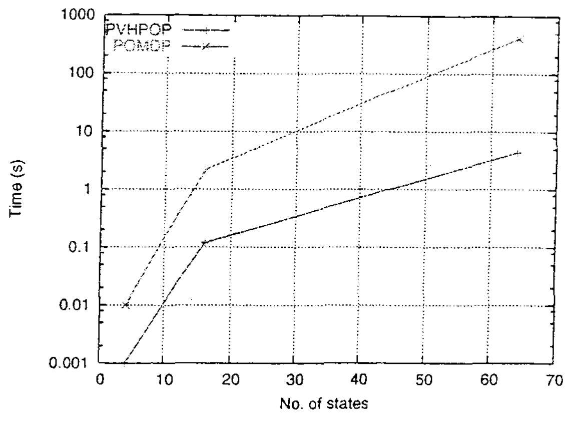

Figure 5: POMDP vs PVHPOP (threshold)

solving the optimal version of the CPP and outperforms CP-Plan in both the SlipperyGripper and SandCastle (Majercik & Littman 1998) domains.

Comparison of the PG heuristics

PVHPOP was provided a range of problems of increasing size and the 2 different heuristics were used. The probability threshold was fixed at 0.9. As can be seen from Figure 3, the additive heuristic performs significantly better than the ratio heuristic. One explanation could be that the emphasis on probability or cost can be for finely controlled by modifying the weight in the additive heuristic. Note that the Y axis is in logarithmic scale.

The ratio heuristic on the other hand seemed to perform reasonably well in experiments where the probability was varied from 0.5. to 0.99 as can be seen in Figure 4. Each curve is represents the computation time as a function of the probability threshold. As the threshold increases, the computation time increases eventually becoming too large as the threshold becomes close to 1.0

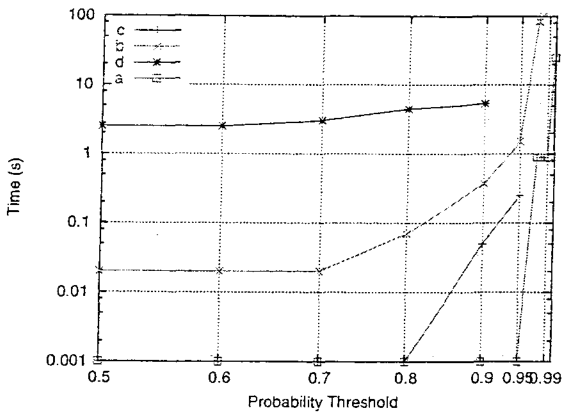

Figure 4: Ratio heuristic over a range of probability thresholds

Comparison with POMDP

In order to compare PVHPOP with the POMDP solver, the problems encoded in PDDL were translated into the state-space representation used by the POMDP solver. This representation consists of the set of states, actions and transition probabilities when a particular action is executed in a particular state. The POMDP solver was configured to assign reward of 0 to every stage except the final stage and was assigned a discount factor of 1.0. The actions were all grounded based on the number of parts in the domain. Three problems consisting of 1,2 and 3 parts were used. In the state-space representation, they correspond to 4, 16 and 64 states. The POMDP solver was run for different horizons and the probability of the goal state was recorded. This probability was set as the probability threshold for PVHPOP and the additive heuristic was used.

As can be seen from Figure 5, PVHPOP scales extremely well compared to the POMDP solver. The plan-graph based additive heuristic is able to effectively guide the planner to select plans that evaluate to a probability greater than the threshold. This performance is similar to the result derived by CAltAlt (Bryce & Kambhampati 2003) for conformant planning problems (non-probabilistic).

The POMDP solver looks for every plan of a particular length(the horizon) and uses the Bellman Optimality principle to generate the optimal k length plan from the optimal k-1 length plan. In each stage of its calculation, it eliminates actions that cannot be part of any optimal plan. However, since the state-space grows exponentially in the number of parts in the domain, the benefit of the optimality principle is overwhelmed by the sheer size of the space. This POMDP solver does not incorporate reachability analysis or abstraction techniques that can reduce the state-space.

An obvious question to ask is how does PVHPOP perform when the probability threshold is increased such that it gets closer to the optimal. Figure 6 shows such a comparison where the probability threshold was increased in step with increasing the horizon of the POMDP solver. The domain used was the SlipperyGripper because it was small domain.

As the threshold gets higher, the performance of PVHPOP deteriorates and eventually becomes worse than the POMDP

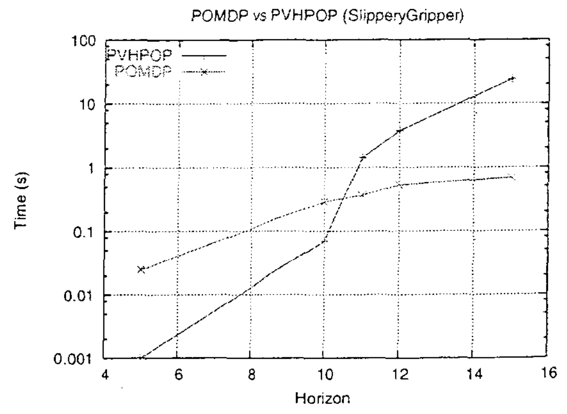

Figure 6: POMDP vs PVHPOP (optimal plan)

solver. This is because PVHPOP has to search many more plan candidates in order to find one that is above the high probability threshold. This result is similar to that identified in (Hyfil & Bacchus 2003). We use a similar argument in that the Bellman principle and the development of a k step plan from the best k-1 step plan significantly enhances the performance of the POMDP solver. PVHPOP can perhaps benefit from a beam search (where not all the children are evaluated) and caching of previously identified good plans for further enhancement.

Discussion

Probabilistic POCL planning guided by plan-graph based heuristics performs very well in solving the variant of the conformant probabilistic planning problem where the objective is to find a plan with a probability of goal achievement above a threshold. This technique also scales very well to larger problems. The heuristics used, though very simple, were able to efficiently guide the planner to find suitable plans. These results mirror the performance of plan-graph based heuristics in both the deterministic planning and non-probabilistic conformant planning areas. However, this technique as implemented has some limitations when asked to find very close to optimal plans.

The CPP problem as defined in this paper requires the planner to find a sequence of actions whose probability of goal achievement is greater than a threshold. In order to do this, PVHPOP constructively searches the space of plans, expanding plans selected based on the plan-graph heuristic. Since there is no requirement to find the optimal plan, it is not necessary to work on every possible partial plan to prove that the plan found is optimal.

State-space based techniques like the POMDP inherently are meant to be used to find the optimal plan. They keep track of each state and the transition from each state to others during every plan stage, pruning clearly dominated actions and using the compact representation of the alpha vectors to keep track of which action is optimal in which belief state. The most significant drawback that these techniques suffer from is the exponential blow up of the state space with respect to the problem size. Current and ongoing research has focused on addressing this drawback using the techniques of abstraction and reachability analysis.

Plan-graphs provide a abstract representation of the state-space as well as allow the performance of reachability analysis. Hence the POCL planning with plan-graph heuristics combine the scalability from searching in the plan-space with the efficiency of a better heuristic.

Future Work

While PVHPOP performs well in terms of computational time, it does consume significant amount of memory since it keeps all the partial plans. Implementing a beam search of size n where only the top n partial plans are stored will significantly reduce the memory requirements. Another area of research is to evaluate additional heuristics. Heuristics based on a relaxed plan derived from a plan-graph have shown promise in deterministic conformant planning.

There have been several choices made in PVHPOP that could potentially have a significant impact on performance if they were changed. For example, it would be interesting to study the impact of reopening supported open conditions after a plan is complete instead of continuing to keep the supported open conditions as open during the planning. Another improvement could be to perform the probability assessment of partial plans incrementally by caching the evaluation of the parent plan. A third aspect is the use of lifted actions instead of grounding them.

Assuming independence of actions, which in many real domains is not true, has a significant impact on the probability assessment. Therefore, an important advance would be to study the issue of retracting this assumption.

A significant question that has not yet been addressed by this research is the impact of flaw selection on the plan. Selecting flaws based on the impact they may have on the overall probability may significantly increase the performance and the quality of the plans produced. For example, it may be useful to only reopen flaws that are on the critical path to the goal rather than choosing a flaw based on the number of refinements or the amount of effort involved.

Acknowledgments

We would like to acknowledge the support of the NASA Intelligent Systems Program for their support. We would also like to thank Nathanael Hyafil for help with using the POMDP solver.

References

Bertoli, P.; Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2001. Mbp: a model based planner.

Blum, A., and Langford, J. 1999. Probabilistic planning in the graphplan framework. In *ECP*, 319–332.

Blythe, J. 1998. *Planning under Uncertainty in Dynamic Domains*. Ph.D. Dissertation, Carnegie Mellon University.

Bryce, D., and Kambhampati, S. 2003. Planning graph heuristics for scaling up conformant planning. In *ICAPS-03 workshop on Planning under Uncertainty and Incomplete Information*, 18–27.

Cassandra, A. pomdp-solve. http://www.cs.brown.edu/research /ai/pomdp/code/index.html.

Draper, D.; Hanks, S.; and Weld, D. 1994. Probabilistic planning with information gathering and contingent execution. In Hammond, K., ed., *Proceedings of the Second International Conference on AI Planning Systems*, 31–36. Menlo Park, California: American Association for Artificial Intelligence.

Hyfil, N., and Bacchus, F. 2003. Conformant probabilistic planning via csps. In *ICAPS-03*, 205–214.

Kushmerick, N.; Hanks, S.; and Weld, D. S. 1995. An algorithm for probabilistic planning. *Artificial Intelligence* 76(1-2):239–286.

Majercik, S. M., and Littman, M. L. 1998. MAXPLAN: A new approach to probabilistic planning. In *Artificial Intelligence Planning Systems*, 86–93.

Onder, N., and Pollack, M. E. 1999. Conditional, probabilistic planning: A unifying algorithm and effective search control mechanisms. In *AAAI/IAAI*, 577–584.

Simon, H. A., and Kadane, J. B. 1975. Optimal problems-solving search: All-or-none solutions. *Artificial Intelligence* 6:235–247.

Smith, D., and Weld, D. 1998. Conformant graphplan. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 889–896.

Younes, H. L. S., and Simmons, R. G. 2002. On the role of ground actions in refinement planning. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling Systems*, 54–61.